Justifying Goal Models

Ivan J. Jureta
Info. Manag. Research Unit,
University of Namur, Belgium
iju@info.fundp.ac.be

Stéphane Faulkner
Info. Manag. Research Unit,
University of Namur, Belgium
stephane.faulkner@fundp.ac.be

Pierre-Yves Schobbens
Institut d'Informatique
University of Namur, Belgium
pys@info.fundp.ac.be

Abstract

Representation and reasoning about information system (IS) requirements is facilitated with the use of goal models to describe the desired and undesired IS behaviors. One difficulty in building and using goal models is in knowing why *a model instance is as it is at some point of the requirements engineering (RE) process. If justifications for modeling choices are missing, an instance of a goal model can neither be considered appropriate nor inappropriate in a given RE project.*

This paper suggests a Goal Argumentation Method (GAM) for recording the decision-making process which results in modeling choices. GAM combines a design rationale approach that guides commonsense reasoning about the goal model with an argumentation model which records and allows analysis of the justification processes leading to modeling decisions.

1. Introduction

A starting point in RE is the representation of goals that the future IS will need to achieve once developed and deployed [17]. Goal modeling can be defined as the activity of representing and reasoning about IS goals using models, in which goals are related through relationships among themselves and with other model elements, such as tasks that system agents/actors are expected to execute, resources that they can use, or roles that they can occupy. With a number of currently established RE methods relying on goal models in the early stages of requirements analysis (e.g., [1], [6], [7], [11], [17], [19]), there seems to be a consensus that such models are useful in RE activities.

When an instance of a goal model is constructed by few stakeholders having similar backgrounds, during a very limited amount of time, and for a relatively simple system, there is no need to record the details of the decision process that has led to the final goal model. However, when a number of stakeholders with different backgrounds participate in RE goal modeling activities, and the rationale behind the modeling decisions is not recorded, the following issues can appear:

- A stakeholder cannot know why another stakeholder has made some modeling decision. The result may be an unnecessary review of the model instance, changes, or additional explaining. These activities require time and resources that could be used in other, more productive tasks.
- A stakeholder cannot recall the reasons for making a modeling decision. While goal modeling is an iterative process, authors' experience indicates that a modeler will often review prior decisions because of imperfect recall of reasons leading to them in the first place. Future iterations could be better informed if arguments behind prior ones are explicit.
- The ideas, arguments, and assumptions underlying a decision remain implicit or are lost over time. Alternative ideas and confronting views that could lead to different, possibly more adequate modeling choices are lost as well. Both can lead to a poor understanding of the problem and of potential solutions. Empirical results suggest that this is an important cause of RE project failure [5].

One possible approach to reducing these problems during RE is to *externalize and document arguments that justify modeling decisions in an instance of a goal model*. To facilitate this task, the paper proposes the Goal Argumentation Method (GAM). The method draws on design rationale approaches [12] and AI argumentation models [2]. Overall, stakeholders construct arguments to justify a modeling decision, or use arguments to question existing arguments and decisions. In this paper, GAM is applied to the goal model from the Tropos method [1], [8], mainly because it uses modeling concepts common in RE. The meeting scheduler case study [18] is used for illustration.

The goal modeling justification problem is first illustrated via an example (§2). The same example is then used to illustrate the features and use of GAM (§3). Related work is discussed (§4), conclusions are drawn, and directions for future work are identified (§5).

14th IEEE International Requirements Engineering Conference (RE'06)
0-7695-2555-5/06 $20.00 © 2006 IEEE

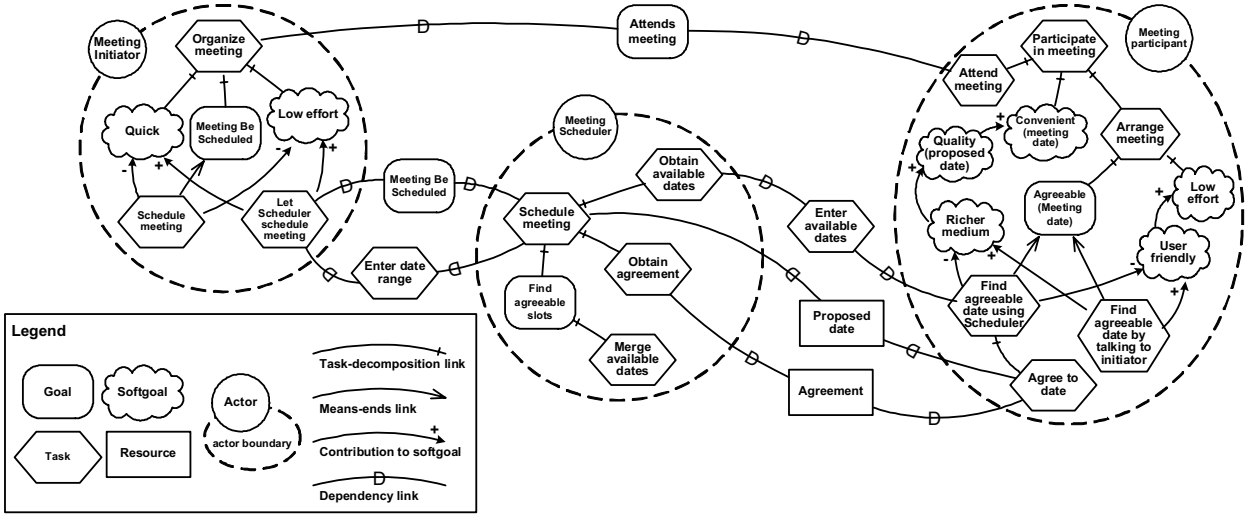


Figure 1. An i-star Strategic Relationship diagram from [19] for the case study

2. The justification problem illustrated

Consider a system for scheduling meetings, similar to that described in [18]. The meeting scheduler should try to select a convenient date and location, such that most potential participants participate effectively. Each meeting participant should provide acceptable and unacceptable meeting dates based on his/her agenda. The scheduler will suggest a meeting date that falls in as many acceptable date sets as possible, and is not in unacceptable date sets. The potential participants will agree on a meeting date once an acceptable date is suggested by the scheduler.

A goal model instance for such a system would be represented in Tropos as an instance of the i-star Strategic Rationale (SR) model[1]. An example SR diagram[2] for such a system, taken as-is from [19], is given in Fig. 1. It shows actors such as *Meeting Scheduler* and *Meeting Participant*, their interdependencies in the achievement of goals, the execution of tasks, and the use of resources, and their internal rationale when participating in the given IS. For example, the *Meeting Be Scheduled* goal of the *Meeting Initiator* can be achieved (represented via a means-ends link) by scheduling meetings in a certain way, consisting of (represented via task-decomposition links): obtaining availability dates from participants, finding a suitable date (and time) slot, proposing a meeting date, and obtaining agreement from the participants. Rectangles with an irregular shape designate softgoals which differ from goals in that there are no objective criteria for their satisfaction. They are commonly used to introduce nonfunctional requirements in a goal diagram.

During the RE process, *the appropriateness of this goal diagram[3] can be evaluated on the basis of arguments that support its given structure*. If arguments are missing, many alternative diagrams that could be produced by the various stakeholders for the same IS could be considered appropriate, provided that they make no errors when using the syntax and semantics of the goal model. While the SR in Fig. 1 serves as a valuable example to illustrate the syntax and semantics of the i-star framework in [19], it is difficult to accept without justification that diagram as more appropriate than another one in a RE project. Lack of arguments supporting the diagram in Fig. 1 can lead a stakeholder reading it to ask, among other the following:

- How does the initiator inform participants that a meeting is being organized?
- Would it not be user friendly for the meeting initiator to inform participants about the meeting using the meeting scheduler?
- Would it not be user friendly if the scheduler looked available dates up in participants' electronic agendas?
- Does the scheduler remind participants of the meeting date? If yes, how/when does it do so? If no, why not?

If justification was explicitly given for the above diagram, all stakeholders might know that, e.g., the initiator prefers to inform participants verbally, that different formats of electronic agendas make it costly to develop a scheduler that can communicate with each participant's software, and so on. Even if such questions are not asked, making it unnecessary for the re-

[1] Because SR extends the basic syntax and semantics of the i-star Strategic Dependency model [19], the SR will be used as the reference Tropos goal model in this paper.

[2] For simplicity, the term *diagram* will be used to replace the term *model instance* in the remainder of the paper.

[3] The appropriateness of a model is defined here as the probability of the IS resulting from the given model to satisfy stakeholder needs. In this paper the focus is on working to increase this probability, not on measuring it.

0-7695-2555-5/06 $20.00

quirements engineer to address them, additional ideas and assumptions that could have surfaced and led to additional requirements as a result of the questions would remain hidden. It would be easier to consider the goal diagram in Fig. 1 appropriate if modeling decisions leading to it are justified.

Although justification can be informal and without a particular structure, the following benefits can be gained by using a structured method:

- By using a design rationale approach [12], each element introduced in a goal diagram can be related to a set of arguments that do or do not justify the modeling decision leading to the given representation of that element. Design rationale approaches assist in commonsense reasoning by organizing a design (here, goal modeling) activity. They can allow the requirements engineer to identify additional requirements from arguments, helping therefore in the modeling activity.
- If arguments are formalized, the justification process can be analyzed for conflict and preference over arguments can be established [2]. Furthermore, the justification process for a goal diagram can be automated.
- When new information becomes available, the change of the goal diagram that it may require can be easier to understand if prior arguments are explicit.

3. The Goal Argumentation Method for justifying goal modeling decisions

To realize the benefits identified in §2, GAM combines a *design rationale* approach and an *argumentation model*. The resulting method is applied when building a goal model to justify modeling decisions, or when questioning already made modeling decisions.

3.1. The design rationale approach in GAM

Design rationale research [12] is concerned with assisting humans when reasoning about the rationale behind decisions that lead to the production of an artifact. A design rationale expresses elements of the reasoning which has been invested behind the design of the artifact [14]. The various design rationale approaches that have been suggested in the software engineering literature give a set of concepts and suggest ways in which they can be manipulated during a design activity (for an overview, see [12]). For example, the IBIS [4] approach consists of relating *issues* that need to be deliberated to *positions* that resolve issues, and *arguments*, that support or object to positions. Recently, [12] suggested the *reasoning loop model*, which integrates common characteristics of established design rationale approaches. It starts from a description of a problem which generates *goals* that characterize potential solutions. Then, *hypotheses* about potential solutions that satisfy goals are generated through problem analysis. Evaluation of alternative hypotheses leads to a *justification* of a selected alternative, which in turn leads to *deciding an action*. The result of an action is likely to lead to new goals, thus restarting the reasoning loop.

The purpose of a design rationale in GAM is to give an overall organization to the decision-making process involved in modeling goals. It clarifies the order of activities that stakeholders need to execute when constructing a goal diagram in such way that existing goal analysis techniques can fit in the modeling process. This way, the design rationale approach fills the gap between abstract suggestions on how the goal modeling activity should be organized (e.g., elicit goals from available documentation and look further by asking *why* and *how* questions [17]) and very specific, low-level analyses already available in RE methodologies (such as goal refinement, abstraction, and operationalization [9], [17]), without requiring any of these to be changed to fit the design rationale approach.

To the best of the authors' knowledge, there are no comparative studies that classify design rationale approaches according to their relevance to goal-oriented RE. As the reasoning loop model claims to be generic, in that it features static and dynamic characteristics common to most approaches (i.e., placing no restriction on the properties of the artifact being designed, being lightweight and informal, and aiming for nonintrusiveness and nonprescriptiveness), it is used in GAM.

However, the original reasoning loop model has been specialized for use in GAM because the latter is intended for use mainly with goal models. Each of the concepts in the reasoning loop model is adjusted to fit the context in which GAM is used. The adapted reasoning loop model is given in Fig. 2.

Goal model construction starts here with a *problem setting*, which generates *problem statements* (called *goals* in the original reasoning loop model[4]). A problem statement designates any objective to be reached, demand to be satisfied, problem to be solved, issue to be discussed, in general anything that one would like to achieve [12]. It should not be mistaken for a RE goal: a problem statement may result in adding a goal to a diagram, but it may also lead to adding any other modeling element or changing the diagram in any other way. *Problem analysis* consists of stakeholders arriving at suggestions, proposals, or ideas about the resolution of the stated problems. The latter are called *alternatives*. These are *evaluated* by generating arguments

[4] This change has been made to avoid misunderstanding, since *goal* in the original reasoning loop model has different meaning than *goal* in RE goal models.

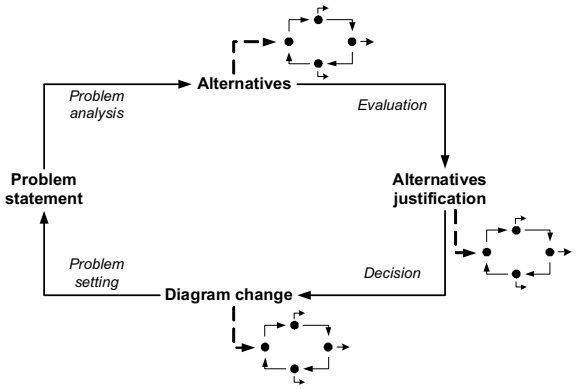


Figure 2. GAM-specific reasoning loop model

that provide support or argue against each alternative. Evaluation leads the stakeholders to accept one alternative by *justifying* their *decision*. Finally, the goal *diagram is changed* to reflect the decision that has been made. The reasoning loop model therefore involves abduction (moving from the problem to alternative solutions) and retroduction (moving from alternatives to the selection and application of an alternative). The completion of a cycle leads to the initiation of a new reasoning loop, until stakeholders agree that no further reasoning about the goal model is required. The reasoning loop must be closed, i.e., all activities need to be executed – otherwise, the stated problem is not adequately addressed.

Notice the dashed arrows. They indicate that additional reasoning loops may be initiated from each of the specific reasoning activities. It is possible to identify new problem statements from alternatives (e.g., an alternative may require existing diagram elements to change in a way not anticipated in the problem statement), alternatives justifications (e.g., new information appearing during evaluation can result in diagram modification), and diagram changes (e.g., adding a new goal may require determining the agent/actor that will be responsible for the achievement of that goal). There is no dashed arrow from the problem statement since it initiates the main reasoning loop again.

Example. To illustrate the use of the reasoning loop in GAM, part of an SR for the meeting scheduler is constructed starting from an empty diagram. The output of the reasoning loop (i.e., the reasoning diagram) is on the left-hand side of Fig.3. The right-hand side of Fig. 3 shows the partial SR associated to the reasoning diagram. To relate SR diagram elements to the elements of the reasoning diagram, each SR element is annotated with the reference of the reasoning loop element from which it is derived. For example, the goal *Schedule Meeting* is marked with *PS1* indicating the reasoning element (here, the problem statement at the root of the reasoning model) that led to the introduction of the goal in the goal model. As a convention, arguments for an alternative are marked with *ArgX+/-*, where X is the number of the argument in the argument list, and + (plus) and – (minus) symbols are used, respectively to indicate that the argument supports or does not support an alternative. Additional observations can be made:

- There is information in the reasoning diagram that is not in the goal diagram (e.g., the reasoning behind the construction of the goal diagram), and there is information in the goal diagram that is not in the reasoning diagram (e.g., that some expression in an argument is a task or a goal). The two diagrams are complementary, allowing a stakeholder to discover why a goal diagram has been constructed in a particular way by reading it in conjunction with the reasoning diagram.
- Because the construction of the incomplete diagrams in Fig. 3 started from an empty sheet, the information in both of them is imprecise, and can be ambiguous: for example, one could ask whether there are security and organizational structure issues (e.g., only a manager can initiate a meeting) in allowing every user to be able to initiate a meeting. Ambiguity is not a problem at this stage, since it is its elucidation that will result in enriching both diagrams and making them precise: For example, one could ask if the task marked *2.Alt1* is a decomposition of another model element, and if so, which one; or if goal *Schedule Meeting* could be decomposed further and how, etc.
- GAM can be used to document the reasoning behind the use of specific goal analysis techniques, already established in RE (such as, e.g., goal refinement and operationalization [17]). For example, the modeler can document reasons leading to, e.g., a refinement of the *Schedule Meeting* goal. In addition, the instance of the reasoning model can also point to the need for refinement or the application of another available technique.
- GAM can be employed, in the same way as above, to raise questions, critique, and structure discussion about a goal model for which the rationale and arguments have not been recorded.

The example above does not use the argumentation model available in GAM. Arguments are written informally, without structure, and there are no particular analysis techniques for argument inconsistency and defeat. Using GAM without a structured approach to arguments and justification is referred to as *light mode*. To relate the content of arguments closer to the content of a goal diagram, and allow analyses such as argument inconsistency, defeat, and counter-argumentation, a formal model of argument is combined to the reasoning loop. GAM *normal* and *advanced* modes, illustrated respectively in §3.2.2 and §3.2.3 use the formal argumentation model.

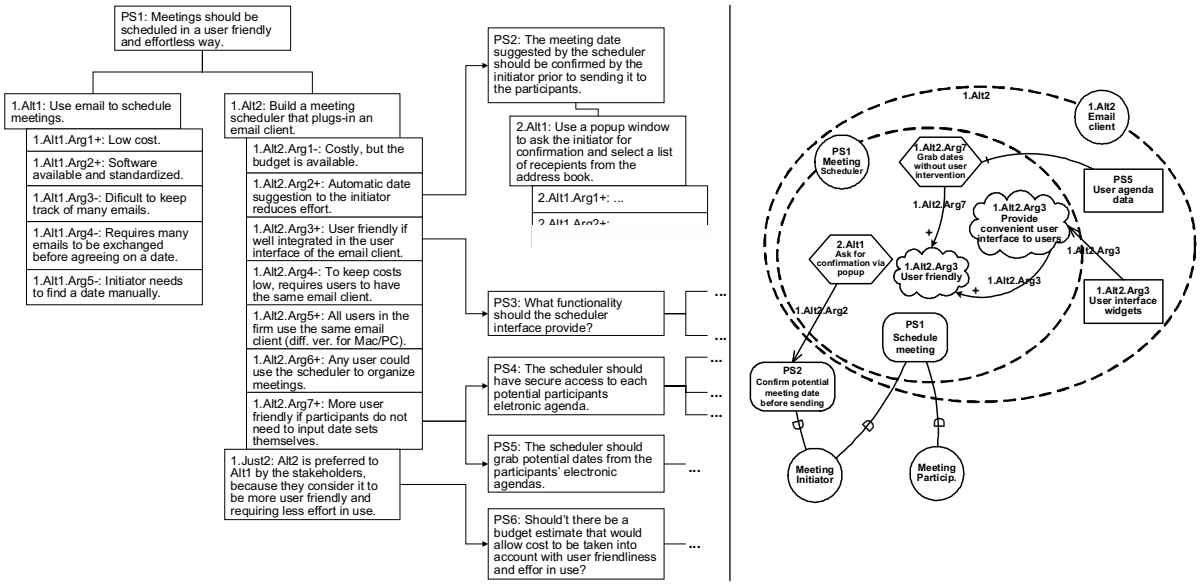


Figure 3. An example of using the GAM reasoning loop to build an i-star SR model

3.2. The argumentation model in GAM

Argumentation modeling literature [2] in the artificial intelligence field focuses on formalizing commonsense reasoning in the aim of automation. An *argumentation model* is a static representation of an *argumentation process*, which can be seen as a search for arguments, where an *argument* is a set of rules that are chained to reach a conclusion. Each rule can be rebutted by another rule based on new information. To formalize *defeasible* reasoning, elaborate syntax and semantics have been developed (e.g., [2], [15]) commonly involving a mathematical logic to formally represent the argumentation process and reason about argument interaction.

The argumentation model in GAM provides a way of structuring the justification process in the *Evaluation* step of the GAM reasoning model and relating the content of arguments closer to the content of a goal diagram.

To illustrate the need for an argumentation model in GAM, consider the example in §3.1 (Fig. 3). Notice that alternative 2 has been chosen over alternative 1. The requirements engineer could alleviate the justification problem by pointing to the fact that all stakeholders agreed that *Alt2* is more adequate than *Alt1* in terms of IS ease of use and user friendliness (*1.Just2*). However, this choice could be considered as *justified* only because the clearly negative argument in *Alt2* (i.e., *1.Alt2.Arg4-*) is rebutted by the argument *1.Alt2.Arg5+*, while the other negative argument *1.Alt2.Arg1-* is written in such way that its second part provides support against its first part. Therefore, the engineer could overlook the ambiguity in *1.Alt2.Arg1-* and conclude that there are no arguments that interfere with alternative 2. In contrast, because there are no arguments that interfere with the negative arguments which themselves interfere with alternative 1 (i.e., *1.Alt1.Arg3-* to *1.Alt1.Arg5-*), choosing alternative 1 is not justified. In presence of an argumentation model, it is required that the arguments for each alternative be more precise and their interrelationships explicit to allow more rigor in justification.

The argumentation model is used in two ways in GAM: a formal unrestricted way, in which the defeasible rules involved in the argumentation process are represented using well-formed formulas (wffs) of a first-order language; and a formal restricted way, in which the defeasible rules are represented with an additional set of predefined keywords based on the ontology underlying the goal model and which are applied on wffs. When GAM is used in the former way, it is referred to as its *normal mode*, whereas the latter is its *advanced mode*. The *normal* mode is suggested when it is preferred that the content of the GAM reasoning diagram be precisely related to the content of the goal diagram, while GAM *advanced* serves as a theoretical basis in providing tool support for recording and analysis of justification for goal modeling decisions.

In contrast, the use of the *light* mode versus the *normal/advanced* modes is dictated by the characteristics of the IS for which the requirements are being engineered (e.g., it may be desirable to use the normal/advanced GAM modes when justifying IS requirements likely to affect system safety or security), the availability of the required expertise (i.e., normal/advanced modes involve the manipulation of for-

malisms), and resource availability (e.g., less time is needed when using the light mode).

The argumentation process is first formally characterized below. The given characterization serves as a basis for GAM *normal* and *advanced mode* uses illustrated later using the running example.

3.2.1. A formal model of argument. This section outlines specific features of the argumentation model suggested in [15] and later extended to avoid fallacious argumentation [16]. This model has been selected over other (for an overview, see [2]) for its integration of prior theoretical ideas in a precise, formal, and usable approach to argument-based defeasible reasoning. The GAM argumentation model is a particular case of this general model applied here in RE activities.

Let $\mathcal{K}$ represent *indefeasible* knowledge that is consistent ($\mathcal{K} \nvdash false$), and let $\mathcal{K}_N$ and $\mathcal{K}_C$ be, respectively *necessary* (general) knowledge and *contingent* (particular) knowledge. Contingent knowledge contains information that depends on the individual constants of a first-order language $\mathcal{L}$ (i.e., depends on the context, or, in other words, this information is expressed with well-formed formulas (wffs) that contain variables), while necessary knowledge is independent of the context. The knowledge of a stakeholder j is then expressed by a pair (K_j, Δ_j), where K_j is a consistent subset of $\mathcal{K}$, and Δ_j is a finite set of *defeasible rules*. A *defeasible rule* has the form $\alpha \succ\!\!-\beta$. The relation "$\succ\!\!-$" between wffs α and β of $\mathcal{L}$ is understood as expressing that "reasons to believe in antecedent α provide reasons to believe in the consequent β". A shorter way to read it is "α is reason for β" [2].

Given the stakeholders' knowledge, it is necessary to determine which of the knowledge that they provide during the reasoning process is *justified* in the given context. To do so, formulas in K ($K = \cup_j K_j$) are related with instances of Δ ($\Delta = \cup_j \Delta_j$) grounded in the given context with a formula h using the *defeasible consequence* "$\mid\!\sim$", defined as follows.

Let A be a member of K and $\Gamma = \{A_1, \ldots, A_n\}$ a set where each A_i is a member of K or is a context-grounded instance of a member of Δ. A well-defined formula A will be called a *defeasible consequence* of the set Γ (i.e., $\Gamma \mid\!\sim A$) if and only if there exists a sequence $B_1, \ldots, B_m$ such that $A = B_m$, and, for each i, either B_i is an axiom of $\mathcal{L}$ or B_i is in Γ, or B_i is a direct consequence of the preceding members of the sequence using modus ponens or instantiation of a universally quantified sentence.

The defeasible consequence is used to define the argument concept: Given a context K and a set Δ, the set T which is a subset of context-grounded defeasible rules is called an *argument* for $h \in \mathcal{K}_C$ in the context K, denoted $\langle T, h\rangle_K$ (or simply $\langle T, h\rangle$), if and only if:

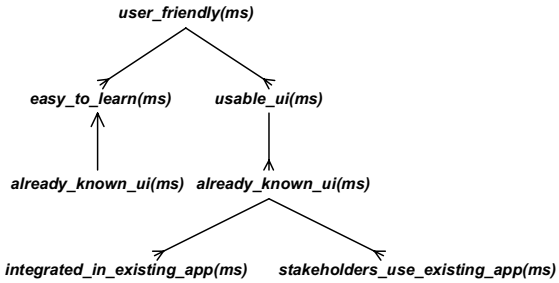


Figure 4. An argument structure

1. $K \cup T \mid\!\sim h$, (K *and* T *derive* h)
2. $K \cup T \nvdash false$ (K, T *are consistent*)
3. $\nexists\, T' \subset T, K \cup T' \mid\!\sim h$ (T *is minimal for* K)

A *subargument* $\langle S, z\rangle$ of $\langle T, h\rangle$ is an argument s.t. $S \subseteq T$.

While an argument can be constructed by combining explicitly expressed knowledge (e.g., from a knowledge base, as is often the case in argumentation modeling literature), the aim with GAM is to start from a sentence h and build arguments that support h from knowledge that stakeholders provide, and that can be related to h. The following example illustrates how the definitions above are used to build an argument in GAM.

Example. Consider the suggestion (see, Fig 3): "1.*Alt2. Arg3+: (Meeting scheduler is) user friendly if well integrated in the user interface of the email client.*" For a meeting scheduler *ms*, a wff h for which we wish to argue or interfere can be written: *user_friendly(ms)*. Stakeholders may then suggest a set of defeasible rules:

Δ = { *easy_to_learn*(X) $\wedge$ *usable_ui*(X) $\succ\!\!-$ *user_friendly*(X), *standardized_ui*(X) $\vee$ *already_known_ui*(X) $\succ\!\!-$ *usable_ui*(X), *integrated_in_existing_app*(X) $\wedge$ *stakeholders_use_existing_app*(X) $\succ\!\!-$ *already_known_ui*(X) }

And the following necessary knowledge (or, knowledge that all stakeholders accept at face value), where $\rightarrow$ is the standard implication operator:

K = { *already_known_ui*(X) $\rightarrow$ *easy_to_learn*(X) }

An argument structure that supports h can be constructed using the given knowledge and defeasible rules, and represented using a tree-like structure as shown in Fig. 4. The root of the tree is the sentence for which the argument structure provides support. To build an argument that interferes with *user_friendly(ms)*, an argument that supports ¬*user_friendly(ms)* is built. Four binary relationships are defined:

- $\langle T_1, h_1\rangle \bowtie \langle T_2, h_2\rangle$: Arguments $\langle T_1, h_1\rangle$ and $\langle T_2, h_2\rangle$ *disagree* iff they are inconsistent for K, i.e., $K \cup \{h_1, h_2\} \vdash false$.
- $\langle T_1, h_1\rangle \xrightarrow{h} \langle T_2, h_2\rangle$: $\langle T_1, h_1\rangle$ *counterargues* $\langle T_2, h_2\rangle$ at h iff a subargument $\langle T, h\rangle$ of $\langle T_2, h_2\rangle$ is such

that $\langle T_1, h_1\rangle$ and $\langle T, h\rangle$ disagree. For example, a subargument $\langle T_3, \neg easy\_to\_learn(ms)\rangle$ of $\langle T_4, \neg low\_effort(ms)\rangle$ counterargues the argument given in Fig. 4.

- $\langle T_1, h_1\rangle \succ_{spec} \langle T_2, h_2\rangle$: Let D be a set of grounded wffs then $\langle T_1, h_1\rangle$ is *strictly more specific* than $\langle T_2, h_2\rangle$ iff: (i) $\forall S \subseteq D$ if $K_G \cup S \cup T_1 \vdash_\sim h_1$ and $K_G \cup S \not\vdash_\sim h_1$ then also $K_G \cup S \cup T_2 \vdash_\sim h_2$. (ii) $\exists S \subseteq D$ such that $K_G \cup S \cup T_2 \vdash_\sim h_2$, $K_G \cup S \not\vdash_\sim h_2$ and $K_G \cup S \cup T_1 \not\vdash_\sim h_1$. The specificity relation establishes a partial order on arguments s.t. more specific arguments are preferred over less specific ones. For example, $\langle \{easy\_to\_learn(ms) \wedge usable\_ui(ms) \succ\!\!- user\_friendly(ms)\}, user\_friendly(ms)\rangle$ is more specific than $\langle \{easy\_to\_learn(ms) \succ\!\!- \neg user\_friendly(ms)\}, \neg user\_friendly(ms)\rangle$, because if *easy_to_learn(ms)* alone is used to activate the argument for $\neg$*user_friendly(ms)*, it cannot by itself activate *user_friendly(ms)*. If *easy_to_learn(ms)* $\wedge$ *usable_ui(ms)* alone is used to activate support for *user_friendly(ms)*, the argument that supports $\neg$*user_friendly(ms)* can also be activated.
- $\langle T_1, h_1\rangle \gg_{def} \langle T_2, h_2\rangle$: $\langle T_1, h_1\rangle$ *defeats* $\langle T_2, h_2\rangle$ iff there is a subargument $\langle T, h\rangle$ of $\langle T_2, h_2\rangle$ such that $\langle T_1, h_1\rangle \xrightarrow{h} \langle T_2, h_2\rangle$, and either $\langle T_1, h_1\rangle \succ_{spec} \langle T, h\rangle$, or $\langle T_1, h_1\rangle$ is unrelated by specificity to $\langle T, h\rangle$.

The *justification process* consists of recursively defining and labeling a *dialectical tree* $\mathcal{T}_{\langle T, h\rangle}$ as follows:

1) A single node containing an argument structure $\langle T, h\rangle$ with no defeaters is by itself a dialectical tree for $\langle T, h\rangle$. This node is also the root of the tree.
2) Suppose that $\langle T_1, h_1\rangle$, …, $\langle T_n, h_n\rangle$ each defeats $\langle T, h\rangle$. Then the dialectical tree $\mathcal{T}_{\langle T, h\rangle}$ for $\langle T, h\rangle$ is built by putting $\langle T, h\rangle$ at the root of the tree and by making this node the parent node of roots of dialectical trees rooted respectively in $\langle T_1, h_1\rangle$, …, $\langle T_n, h_n\rangle$.
3) When the tree has been constructed to a satisfactory extent by recursive application of 1) and 2), label the leaves of the tree *undefeated* (*U*). For any inner node, label it undefeated iff every child of that node is a *defeated* (*D*) node. An inner node will be a defeated node iff it has at least one U node as a child. Do step 4) after the entire tree is labeled.
4) $\langle T, h\rangle$ is a *justification* (or, *T is justification for h*) iff the node $\langle T, h\rangle$ is labeled *U*.

The above argumentation model and justification process are adapted for use in GAM, as follows.

3.2.2. GAM normal mode. When it is preferred to precisely relate the contents of the GAM reasoning diagram to the contents of the goal diagram, to use argument binary relationships to analyze argument interaction, and to use the above justification process, GAM is used in its *normal mode*.

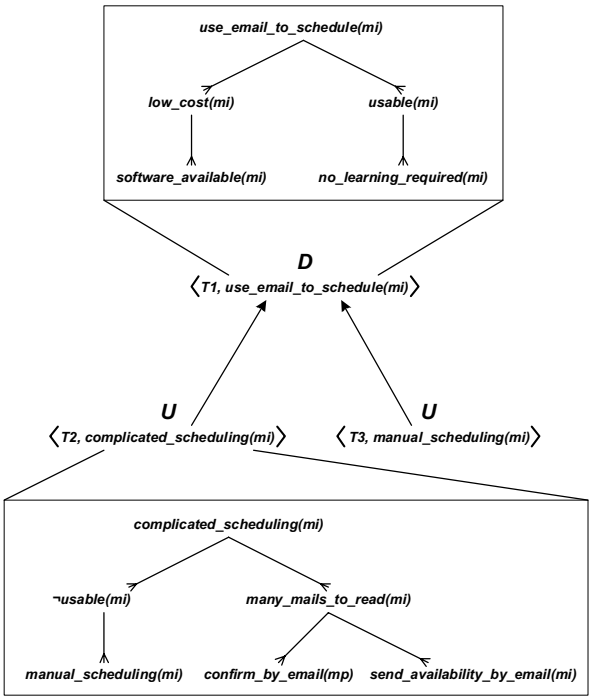


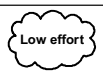
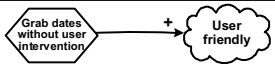
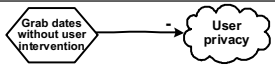
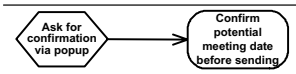
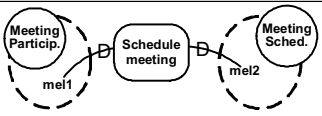
Figure 5. A dialectical tree with detail shown

In normal mode, the reasoning model (§3.1, Fig. 2) is combined with the justification process (§3.2.1) in the following way. First, when *problem analysis* leads to the identification of a set of alternatives, a dialectical tree is built for each alternative using the syntax and semantics from the argumentation model given above (§3.2.1). Second, the *evaluation* reasoning activity consists of labeling each of the dialectic trees, and accepting the one justified alternative (i.e., the alternative whose dialectical tree is such that the root node is labeled as undefeated). *It is required that only one alternative be justified* – in case more than one alternative appear justified, additional arguments need to be added as leaf nodes to each alternative's dialectic tree until only one alternative remains justified. The *decision* reasoning activity is reduced to choosing the alternative that is justified, and acting upon it in terms of changing the associated goal diagram.

Example. The dialectical tree for alternative 1 "*Use email to schedule meetings*" is shown in Fig. 5. Instead of using informal arguments, the dialectical tree is built with wffs. The justification process allowed to show that the alternative is unjustified and therefore cannot be accepted. Argument structures for $\langle T1, use\_email\_to\_schedule(mi)\rangle$ and $\langle T2, complicated\_scheduling(mi)\rangle$ are also shown. The argument $\langle T2, complicated\_scheduling(mi)\rangle$ defeats $\langle T1, use\_email\_to\_schedule(mi)\rangle$ at $\neg usable(mi)$ due to subargument $\langle \{manual\_scheduling(mi) \succ\!\!- \neg usable(mi)\}, \neg usable(mi)\rangle$ of $\langle T2, complicated\_scheduling(mi)\rangle$.

0-7695-2555-5/06 $20.00

Table 1. GAM/Tropos translation rules with examples

Element in the Tropos goal diagram	*Intermediary language*	*Labeled well-formed formula in a dialectical tree*
	*goal(Schedule meeting)*	*achieve(schedule_meeting(ms))*
	*softgoal(Low effort)*	*optimize(schedule_meeting(ms))*
	*task(Ask for confirmation via popup)*	*do(ask_for_confirmation(ms))*
	*resource(User agenda data)*	*provide(user_data_agenda(email_client))*
	*actor(Meeting Participant)*	*meeting_participant* or ***ms*** *(a variable, not a wff)*
	*contribution[+](task(Grab dates withour user intervention), softgoal(user friendly))*	*do(grab_dates_automatically(ms))* *optimize(user_friendly(ms))*
	*contribution[-](task(Grab dates withour user intervention), softgoal(user privacy))*	*do(grab_dates_automatically(ms))* *¬optimize(user_privacy(ms))*
	*task-decomposition(task(Grab dates withour user intervention), resource(User agenda data))*	*provide(user_data_agenda(email_client))* *do(grab_dates_automatically(ms))*
	*means-ends(task(Ask for confirmation via popup), goal(Confirm potential meeting date before sending))*	*do(ask_confirmation_via_popup(ms))* *achieve(confirm_before_sending(mi))*
	*dependency(mel1, goal(schedule_meeting), mel2)* (see GAM/Tropos translation rules for meaning of **mel1** and **mel2**)	(see transl. rules) *depend(mp, achieve(schedule_meeting(ms)), ms)* *achieve(schedule_meeting(ms))* (see transl. rules)

3.2.3. GAM advanced mode. To allow tool support to be provided for automated recording and analysis of arguments for goal modeling decisions, the GAM *advanced mode* is defined. This is realized by defining a mapping allowing the stakeholders to translate between a goal diagram and a dialectical tree. A tool can then be constructed based on these rules. In this paper, rules for translating between a Tropos Goal Diagram (TGD) and a dialectical tree (DT) are suggested. The DT can thus be submitted to Tropos-specific analyses (if it is translated into a TGD), and a TGD can be subjected to the analysis of the justification behind the modeling decision that led to it.

Mapping is defined using an intermediary language and a set of keywords illustrated with case study examples in Table 1. When translating from a TGD to a DT (moving from left to right in Table 1) the intermediary language is used to write down the structure of the TGD. The obtained TGD specification is then translated into *wffs* labeled with a restricted set of keywords. The rules used for translating between the intermediary language and the DT are referred to as the *GAM/Tropos translation rules* and are formalized as follows. The operator "$_{\text{label}}\langle\approx\rangle$" is used to mark translation rules, with *label* indicating the name of the rule being used in the translation:

$goal(Name)\ {}_{\text{goal}}\langle\approx\rangle\ achieve(wff) \mid maintain(wff) \mid achieve\&maintain(wff) \mid avoid(wff)$

$task(Name)\ {}_{\text{task}}\langle\approx\rangle\ do(wff)$
$resource(Name)\ {}_{\text{resource}}\langle\approx\rangle\ use(var) \mid provide(var)$
$softgoal(Name)\ {}_{\text{softgoal}}\langle\approx\rangle\ optimize(wff)$
$actor(Name)\ {}_{\text{actor}}\langle\approx\rangle\ var$

$cgmodel := goal(Name) \mid \ldots \mid softgoal(Name)$
$cgmodeltype := goal \mid \ldots \mid softgoal$
$ckeyword := achieve(wff) \mid maintain(wff) \mid \ldots \mid optimize(wff)$

$contribution[+](cgmodel_1, cgmodel_2)\ {}_{\text{contribute[+]}}\langle\approx\rangle\ (cgmodel_1\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword_1 \wedge cgmodel_2\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword_2 \wedge ckeyword_2 \succ\!\!-\!\!- ckeyword_1)$.

$contribution[-](cgmodel_1, cgmodel_2)\ {}_{\text{contribute[-]}}\langle\approx\rangle\ (cgmodel_1\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword_1 \wedge cgmodel_2\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword_2 \wedge \exists\langle T_1, h_1\rangle, \langle T_2, h_2\rangle$ *such that* h_1 *is the wff in* $ckeyword_1$ *and* h_2 *is the wff in* $ckeyword_2$ *and* $\langle T_2, h_2\rangle \gg_{\text{def}} \langle T_1, h_1\rangle$).

$task\text{-}decomposition(task(Name), cgmodel)\ {}_{\text{task-decomposition}}\langle\approx\rangle\ (cgmodel\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword \wedge task(descr)\ {}_{\text{task}}\langle\approx\rangle\ do(wff) \wedge\ do(wff) \succ\!\!-\!\!- ckeyword)$.

$means\text{-}ends(goal(descr), cgmodel)\ {}_{\text{means-ends}}\langle\approx\rangle\ (\ cgmodel\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword \wedge goal(descr)\ {}_{\text{goal}}\langle\approx\rangle\ achieve(wff) \mid \ldots \mid avoid(wff) \wedge achieve(wff) \mid \ldots \mid avoid(wff) \succ\!\!-\!\!- ckeyword)$.

$dependency(mel_1, cgmodel, mel_2)\ {}_{\text{dependency}}\langle\approx\rangle\ ($
$cgmodel\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword \wedge$
$(\forall i = \{1,2\},\ mel_i = (cgmodel_{i,1}, \ldots, cgmodel_{i,r}) \wedge$
$\forall\ 1 \le k \le r,\ cgmodel_{i,k}\ {}_{\text{cgmodeltype}}\langle\approx\rangle\ ckeyword_{i,k}$
with $cgmodel_{i,1}\ {}_{\text{actor}}\langle\approx\rangle\ var_i \wedge r > 0 \wedge$
$\forall i,\ dep_i = \wedge_{\text{m}} ckeyword_{i,m},\ 2 \le m \le r) \wedge$

0-7695-2555-5/06 $20.00

Figure 6. Meeting scheduler Tropos SR diagram (Fig. 1) translated using GAM/Tropos translation rules

$(ckeyword \wedge dep_1) \succ\!\!- depend(var_1, ckeyword, var_2) \wedge depend(var_1, ckeyword, var_2) \succ\!\!- dep_2)$

Observe that the relationships in the goal model are interpreted as defeasible rules. Intuitively, this seems adequate: e.g., if a task is decomposed into a resource in a TGD, the need to provide the resource can be interpreted to exist because that resource is used when executing the given task; in a negative contribution, the link in the TGD is directed from an element that contributes negatively to the target softgoal, whereas in a DT, a negative contribution exists between a defeater argument and the argument it defeats. The dependency relationship is interpreted as a chain of two defeasible rules ($\alpha \succ\!\!- \beta \succ\!\!- \chi$). In the first, β marks the dependency between actors and α is the dependum of the dependency (because, e.g., in a goal dependency, the dependum goal is a reason for the dependency to exist: the depender cannot achieve the goal without the dependee). In addition, α can contain one or more *ckeyword* to indicate why the depender alone is unable to obtain the dependum (this is required if a Tropos SR is being translated, whereas it is often unknown in a Tropos SD). In $\beta \succ\!\!- \chi$, χ expresses goals/tasks/…/softgoals that the dependee is expected to, respectively achieve/…/optimize in order to assist the depender in obtaining the dependum (χ is often unknown in a Tropos SD, whereas it is available in a Tropos SR). To translate a DT to a TGD, the wffs appearing in the DT are transformed into *labeled* wffs. Labels are used to derive a goal diagram element from a wff. For wffs that are to be translated into goals in a TGD, the Tropos goal taxonomy [8] is employed, giving four labels: *achieve(wff)*, *maintain(wff)*, *achieve&maintain(wff)*, and *avoid(wff)*. For wffs that will result in resources, the label *provide(wff)* is used when a resource is being provided by an actor, whereas the label *use(wff)* is applied when the resource is to be used by an actor. Table 1 and the Tropos/GAM translation rules give other labels along with their corresponding TGD representation.

Example. The dialectical tree whose argument structures are shown in Fig. 6 has been obtained by applying the translation rules to the Tropos SR in Fig. 1. Since the SR in Fig. 1 contained alternatives, one alternative has been translated. Using the obtained dialectical tree, the stakeholders can e.g., question modeling choices by providing new arguments that defeat existing ones and lead to changes in the tree. For example, the labeled rule *optimize(low_effort(mi))* $\succ\!\!-$ *do(let_scheduler_schedule(mi))* may appear inadequate to some stakeholders. They may add additional rules that would either defeat the antecedent or the consequent, or provide further arguments to support both. For lack of space, we do not discuss circular arguments that appear in the above tree and that may be undesirable, as circularity amounts to supporting or denying something by itself. An example is *do(schedule_meeting(ms))* which partially supports itself through other defeasible rules (see bottom of Fig. 6).

0-7695-2555-5/06 $20.00

4. Related work

The goal-oriented RE literature provides mainly two ways to justify the structure of a goal model. The first, based on the NFR method for nonfunctional requirements analysis [3] has been explicitly integrated in Tropos. It consists of defining a set of imprecise nonfunctional goals (i.e., softgoals) which express criteria for system quality (e.g., usability, adaptability, etc.), and evaluating the degree to which an alternative model structure would lead to an IS that is usable, adaptable, etc. Evaluation is qualitative. Argumentation goals are used for justification and are specialized for and limited to softgoal models [13]. In the second method [10], probability of each alternative to satisfy a set of goals is estimated and used for comparing alternatives. While this brings the comparison process closer to objectivity and precision, it appears difficult and costly to use as it requires probability estimates to be produced in some way. Because justification for a model structure is summarized in a probability estimate, information leading to the estimate can remain hidden for most stakeholders.

A third way, explored in this paper, is to allow qualitative and quantitative information into arguments. The novelty of GAM is to combine a design rationale approach to organize commonsense reasoning with a formal argumentation model to document the arguments leading to a modeling decision and allow structured justification to take place, while allowing informal and formal use with any goal model. As arguments accept both qualitative and quantitative information, the importance of quantitative evidence can be acknowledged, along with qualitative, subjective, and defeasible information.

5. Conclusions and future work

A RE goal diagram can be considered neither appropriate nor inappropriate if modeling decisions leading to that diagram are not justified. This paper suggested a Goal Argumentation Method (GAM) used to justify modeling decisions that lead to a specific goal diagram. In a RE project, stakeholders can use GAM in different ways. The *light mode* complements goal diagrams with reasoning diagrams containing unstructured and informally stated arguments. The *normal mode* introduces formal argumentation to allow the requirements engineer to precisely relate the contents of the GAM reasoning diagram to that of the goal diagram, to use binary relationships between arguments to analyze argument interaction within a structured justification process. The *advanced mode* defines RE method-specific mapping between a goal diagram and a dialectical tree so that tool support can be provided for automated recording and analysis of arguments for goal modeling decisions.

Effort is currently focused on three issues. First, we explore whether translation rules independent of the RE method can be defined, due to the fact that at least some conceptual foundations are shared by various established goal-oriented RE methods. Second, we work on developing analysis techniques that would enrich traditional goal-oriented RE in terms of allowing, e.g., fallacious argumentation to be discovered in a goal diagram. Third, we intend to develop a tool on the basis of translation rules in GAM advanced mode.

6. References

[1] J. Castro, M. Kolp, and J. Mylopoulos, "Towards requirements-driven information systems engineering: the Tropos project", *Info. Sys.*, 27(6), 2002, pp. 365-389.

[2] C.I. Chesnevar, A.G. Maguitman, and R.P. Loui, "Logical Models of Argument", *ACM Comp. Surv.*, 32(4), 2000.

[3] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer, 2000.

[4] J. Conklin, and M.L. Begeman, "gIBIS: A hypertext tool for exploratory policy discussion", *ACM Trans. Inf. Syst.*, 6(4), 1988.

[5] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems", *Comm. ACM*, 31(11), 1988.

[6] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition", *Sci. Comp. Progr.*, 20, 1993.

[7] P. Donzelli, "A goal-driven and agent-based requirements engineering framework", *Req. Eng.*, 9(1), 2004, pp.16-39.

[8] A. Fuxman, L. Liu, Mylopoulos J., M. Pistore, M. Roveri, and P. Traverso, "Specifying and Analyzing Early Requirements in Tropos", *Req. Eng.*, 9(2), 2004, pp. 132-150.

[9] E. Letier *Reasoning about Agents in Goal-Oriented Requirements Engineering*. Phd Th., Univ. of Louvain. 2001.

[10] E. Letier, and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering", *ACM Sigsoft Softw. Eng. Notes*, 29(6), Nov. 2004, pp. 53-62.

[11] L. Liu, and E. Yu, "Designing information systems in social context: a goal and scenario modeling approach", *Info. Sys.*, 29, 2004.

[12] P. Louridas, and P. Loucopoulos, "A Generic Model for Reflective Design", *ACM Trans. Softw. Eng. Meth.*, 9(2), 2000.

[13] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", *IEEE Trans. Softw. Eng.*, 18(6), 1992, pp. 483-497.

[14] B.S. Shum, and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?" *Int. Journ. Human-Comp. Studies*, 40(4), Apr. 1994, pp. 603-52.

[15] G.R. Simari, and R.P. Loui, "A mathematical treatment of defeasible reasoning and its implementation", *Artif. Int.*, 53, 1992.

[16] G.R. Simari, C.I. Chesnevar, A.J. Garcia, "The Role of Dialectics in Defeasible Reasoning", *Proc. XIV Int. Conf. Chilean Soc. in Comp. Sci.*, 1994.

[17] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", *Proc. 5th IEEE Int. Symp. Req. Eng.* 2001.

[18] A. Van Lamsweerde, R. Darimont, and Ph. Massonet, "The Meeting Scheduler Problem: Preliminary Definition", Univ. catholique de Louvain.

[19] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", *Proc. 3rd IEEE Int. Symp. Req. Eng.*, 1997.